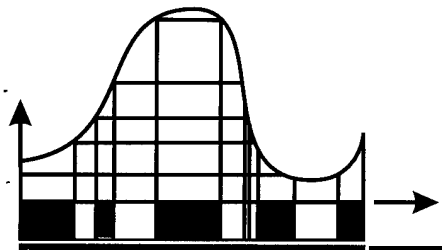

Center for Foundations of Intelligent Systems

Technical Report
98-04

Performance Evaluation of Synchronization Losses in the Continuous Media Toolkit

D. Wijesekera, S. Parikh, S. Varadarajan, J. Srivastava, A. Nerode and M. Foresti

February 1998

CORNELL
UNIVERSITY

625 Rhodes Hall, Ithaca, NY 14853 (607) 255-8005

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	31 March 1998	**Technical Report**

4. TITLE AND SUBTITLE
Performance Evaluation of Media Synchronization Losses in the Continuous Media Toolkit

5. FUNDING NUMBERS
DAAH04-96-1-0341

6. AUTHOR(S)
D. Wijesekera, S. Varadarajan, S. Parikh, J. Srivastava, A. Nerode and M. Foresti

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Regents of the University of California
c/o Sponsored Projects Office
336 Sproul Hall
Berkeley, CA 94720-5940

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
ARO 35873.69-MA-MUR

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by the documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
This paper presents a performance analysis of synchronization services provided by the Berkeley Continuous Media Toolkit (CMT). The quality of audio-video synchronization is measured against processor and network loads for both remote and local clients. The metrics of analysis are the perceptible and tolerable human perceptual limits reported by Steinmetz, and another metric designed to measure synchronization of lossy media streams. It is shown that according to Steinmetz' metric CMT provides imperceptible audio-video mis-synchronization for about 10 seconds, and tolerable synchronization for about 13 seconds from the start of the clips for local clients under low processor loads. It is also shown that under high loads, synchronization is achieved at the cost of losing media frames.

14. SUBJECT TERMS
quality of service, user studies, media losses, metrics, multimedia synchronization

15. NUMBER OF PAGES
19

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	**UNCLASSIFIED**	**UNCLASSIFIED**	**UL**

NSN 7540-01-280-5500

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18
298-102

Performance Evaluation of Synchronization Losses in the Continuous Media Toolkit*

Duminda Wijesekera, Shwetal Parikh, Srivatsan Varadarajan,
Jaideep Srivastava, Anil Nerode† and Mark Foresti‡.
Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.
Institute for Foundations of Intelligent Systems, Cornell University, Ithaca, NY 14853†.
Rome Laboratory, Griffis Air Force Base, Rome, NY 14853‡.
e-mail: {wijesek|ssparikh|varadara|srivasta}@cs.umn.edu,
nerode@hybrid.cornell.edu,forestim@rl.af.mil

Abstract

This paper presents a performance analysis of synchronization services provided by the Berkeley Continuous Media Toolkit (CMT). The quality of audio-video synchronization is measured against processor and network loads for both remote and local clients. The metrics of analysis are the perceptible and tolerable human perceptual limits reported by Steinmetz, and another metric designed to measure synchronization of lossy media streams. It is shown that according to Steinmetz' metric CMT provides imperceptible audio-video mis-synchronization for about 10 seconds, and tolerable synchronization for about 13 seconds from the start of the clips for local clients under low processor loads. It is also shown that under high loads, synchronization is achieved at the cost of losing media frames.

Key Phrases: Quality of Service, Toolkit support for multimedia, Multimedia synchronization, Performance evaluation, Loss measurement

1 Introduction

Rapid growth of multimedia systems, and accordingly research efforts in this area, have made it necessary to have toolkit support for rapid prototyping. Being one of the most popular toolkits offered in multimedia, the *Berkeley Continuous Media Toolkit (CMT)* [SRY93, MPR97] has gone a long way in fulfilling this need. In [MPR97], Mayer-Patel and Rowe measured the performance of CMT and its system overheads. In this paper, we measure the quality of audio-video synchronization obtainable in CMT.

Most multimedia systems display synchronized audio and video. In order to give a *natural* effect, and therefore to provide a sense of cohesiveness and ease of comprehension, audio and video have to be synchronized within human perceptual limits [Ste96, SE93].

CMT provides the *Logical Time System (LTS)*, a mechanism for maintaining the progress of streams and providing inter-stream synchronization [MPR97]. In a series of experiments we investigate the quality of audio-video synchronization provided by the LTS mechanism against known metrics.

*This work is partially supported by Air Force contract number F30602-96-C-0130 to Honeywell Inc, via subcontract number B09030541/AF to the University of Minnesota and DOD MURI grant DAAH04-96-10341 to Cornell University

Since our evaluations measure perceptible quality, they should be measured external to the system that is providing the continuous media (CM) services [SNL95]. As shown in [SNL95], internal and external measurements of synchronization could be different. As a first step in this direction, we report internal measurements, since the quality of the latter can be no better than the former, and consequently we can obtain a lower on externally measured synchronization. Additionally, we are currently in the process of externally measuring the synchronization behavior of CMT.

In our usage, we have noticed that CMT drops frames in its stream services. Therefore, we have chosen two metrics to measure audio-video synchronization. The first one [Ste96] is designed to measure synchronization between loss-less media streams. The second [WS96] has been designed to measure synchronization quality in lossy streams.

Our experiments indicate that for a single site display, CMT provides tolerable audio-video synchronization for about 13 seconds under low processor load, and that this value decreases under higher loads. We have also found that in distributed playing, there is hardly any tolerable synchronization, and that available synchronization decreases as processor or network load increases, calling for additional mechanisms to maintain tolerable audio-video synchronization.

The rest of the paper is organized as follows. Section 2, describes the metrics used to measure synchronization in lossy media. Section 3 is a brief introduction to the structure and functionality of CMT. Section 5 gives the analysis of our experimental outcome with respect to metrics described in [Ste96, SE93], and Section 6 gives their analysis with respect to metrics described in [WS96]. Finally, Section 7 ends the paper with concluding remarks.

2 Synchronization Metrics

For the purpose of describing QoS metrics for lossy media streams, we envision a CM stream as a flow of data units (referred to as logical data units - LDUs in the uniform framework of [SB96]). In our case, we take a video LDU to be a frame, and an audio LDU to constitute 8000/30, i.e. 266 samples of audio[1]. Given a rate for streams consisting of these LDUs, we envision that there is time slot for each LDU to be played out. In the ideal case a LDU should appear at the beginning of its time slot.

2.0.1 Metrics for Continuity

Continuity of a CM stream is metrized by three components: *rate, drift* and *content.* The ideal rate of flow and the maximum permissible deviation from it constitute our *rate* parameters. As stated, given the ideal rate and the beginning time of a CM stream, there is an ideal time for a given LDU to arrive/ be displayed. Given the envisioned fluid-like nature of CM streams, the appearance time of a given LDU may deviate from this ideal. Our *drift* parameters specify aggregate and consecutive non-zero drifts from these ideals, over a given number of consecutive LDUs in a stream. For eg., first four LDUs of two example streams with their expected and actual times of appearance, are shown in Fig. 1. In the first example stream, the drifts are respectively 0.0, 0.2, 0.2 and 0.2 seconds; and accordingly it has an aggregate drift of 0.6 seconds per 4 time slots, and a non-zero consecutive drift of 0.6 seconds. In the second example stream, the largest consecutive non-zero drift is 0.3 seconds and the aggregate drift is 0.5 seconds per 4 time slots. The reason for a lower consecutive drift in stream 2 is that the unit drifts in it are more spread out than those in stream 1.

[1]SunAudio has 8-bit samples at 8kHz, and an audio frame constitutes of 266 such samples equivalent to a play time of one video frame, i.e. 1/30 seconds.

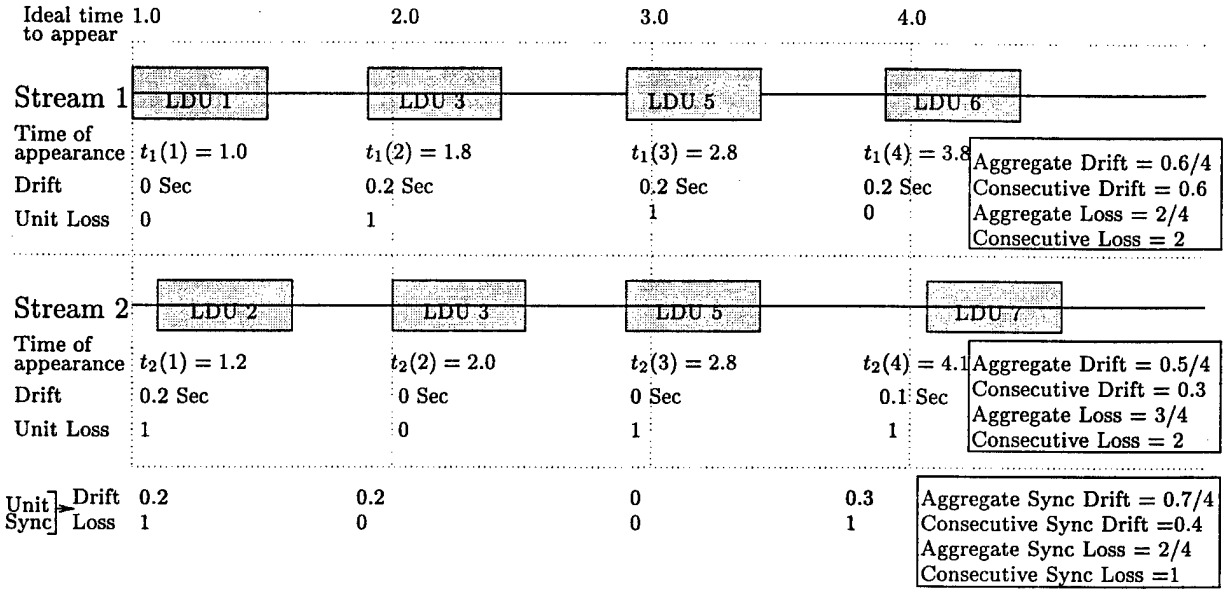

Figure 1: Two Example Streams used to Explain Metrics

In addition to timing and rate, ideal contents of a CM stream are specified by the ideal contents of each LDU. Due to loss, delivery or resource over-load problems, appearance of LDUs may deviate from this ideal, and consequently lead to discontinuity. Our metrics of continuity are designed to measure the average and bursty deviation from the ideal specification. A loss or repetition of a LDU is considered a unit loss in a CM stream. (A more precise definition is given in [WS96].) The aggregate number of such unit losses is the *aggregate loss* of a CM stream, while the largest consecutive non-zero loss is its *consecutive loss*. In the example streams of Fig. 1, stream 1 has an aggregate loss of 2/4 and a consecutive loss of 2, while stream 2 has an aggregate loss of 2/4 and a consecutive loss of 1. Once again, the reason for the lower consecutive loss in stream 2 is that its losses are more spread-out than those of stream 1.

2.0.2 Metrics for Synchronization in Lossy Media

As in the case of continuity metrics, synchronization metrics are also categorized into content, rate and drift. Rate of rendition of a collection of synchronized streams is determined by the rates of component streams. (must be equal to be synchronized) The rate variation of a collection of synchronized streams is taken to be the maximum of their component streams.

In a perfectly synchronized collection of streams, the i^{th} LDU of each stream should start playing out at the same instant of time. Failure to accomplish this ideal is measured by the maximum difference between the display start time of the LDUs in the group, and is referred to as the unit synchronization drift. The aggregate of such unit synchronization drifts over a given number of LDU slots is the aggregate synchronization drift, and the maximum of such non-zero consecutive synchronization drifts is the consecutive synchronization drift. They measure the average and bursty time drifts in synchronization. In Fig. 1, the two streams have unit synchronization drifts of 0.2, 0.2, 0.0, and 0.3 seconds respectively, resulting in an aggregate synchronization drift of 0.7/4, and a consecutive synchronization drift of 0.4 seconds.

For the content component, with streams consisting of LDUs with equal play-out times, there is a natural collection of LDUs that are to be played out simultaneously. The largest discrepancy in the LDU numbers between any two pairs in such a collection is referred to as the unit synchronization

loss. The aggregate and largest non-zero consecutive unit synchronization loss is referred to as *aggregate synchronization content loss* and *consecutive synchronization content loss*, respectively. In the example of Fig. 1, due to losses of LDUs there are unit synchronization content losses at the first and the last pairs of LDU's, resulting in an aggregate synchronization content loss of 2/4 and a consecutive synchronization loss of 1.

2.0.3 Parameters for Lossy Metrics

In a user study [WSNF97] it has been determined that aggregate losses of upto 17/100 were imperceptible, and those beyond 23/100 were annoying, and in between these two values were tolerable. The tolerable value for consecutive losses were determined to be two frames. For audio this limit was about three frames. Due to the inability of precisely introducing drifts, the user study did not estimate any values for drifts.

For audio-video synchronization, 6/100 was determined to be the limit of imperceptible aggregate loss and 7/100 was determined to be the limit for tolerable aggregate loss. A unit consecutive loss (i.e. 1/100) was determined to be the tolerable limit.

3 CMT: Design and Functionality

The Berkeley Continuous Media Toolkit, commonly referred to as CMT, has been constructed for the purposes of application development and research in multimedia systems[MPR97]. It is an extensible system consisting of a three layered architecture. The topmost layer, referred to as the *application code layer*, consists of Tcl, Tk and Tcl-Dp [SRS93] code. The next layer, consists of a CM object model, time and synchronization services, CM event services and storage/buffer services for CM streams. The bottom most *resource layer* consists of operating system and hardware services. The complete design of CMT is explained in detail in [MPR97].

One of the outstanding features of CMT is the ease of programming in Tcl/Tk [Ous94, Wel95], and the extensible and relatively *policy free* nature of implementation, which makes it a great testbed for experimentation.

CMT envisions multimedia as streams flowing in and out of CM objects, in their journey between *sources* and *sinks*. These CM objects correspond to *services* that are used by CM streams, such as *segments*, that represent data sources stored in files. Similarly, *play* objects represent stream players, such as speakers and video displays. CMT provides distributed multimedia services in the sense that objects in a CM pipeline can sit on different locations, thereby requiring network transport services. The network twin objects *packet source (pktSrc)* and *packet destination (pktDest)* provide these services of sending and receiving streamed data.

A CMT application program specifies pipelines of CM streams flowing between CM objects in Tcl/Tk. The objects can be created within some *CM process*. These scripts are interpreted by a Tcl interpreter extended with Tcl-DP to provide distributed services. For an application that uses only one location, a single CM process is created with appropriate sources and sinks. Distributed applications have CM processes at each location, where objects belonging to that location are created within the corresponding process.

The Logical Time System (LTS) in the CMT Library layer provides a mechanism for applications to maintain a concept of where in time the application is, and how fast time is progressing. More than one LTS can exist in one application and different CM processes can be synchronized by using the same LTS. CMT ensures that the LTS progresses according to the *rate* specified by the application program.

CMT maintains CM streams by passing data between objects by either the push model or the pull model. In our experimental setup, we used the push model, where the producer of data, typically the source object, initiates the data transfer to the object immediately downstream from it. The source object maintains a continuous streams of CM data by periodically invoking itself to fetch data from the specified file at the specified rate, and transfers data to the corresponding object downstream. The period of such data fetches are specifiable in application code. Objects that are being called by some other object upstream provides an *accept* method, that is being passed a buffer containing appropriate data by the callee. After *processing* the data in the buffer, such an intermediate object calls the accept method of its immediate downstream neighbor, thereby maintaining the flow of CM data through the specified pipeline, until a sink object such as a display device is reached. The sink objects are the final consumers of CM data, whereby it is displayed to the human viewer.

3.1 Application Code Used in Our Experiment

In our experiments, we use two application programs, referred to as *local playback application* and *remote playback application* in [MPR97], graphically represented in Fig 2. The local experiment consists of a single CM process containing four objects, namely: *mjpegSegment*, *auSegment*, *mjpeg-Play* and *auPlay*. As shown in Fig 2, they are pipelined so that *mjpegSegment* passes a video stream to *mjpegPlay* to be displayed at the local screen and *auSegment* passes an audio stream to *auPlay* to be displayed at the speaker. Although the application program does not explicitly use it, the play objects call an accept method of a low level object called *device*, that is responsible for the final display of media units.

In the *remote play* experiment, the segments and play objects are on two different machines that are connected by Ethernet. Consequently, in addition to the objects used in the local play, *packet source* and *packet destination* network twins are used.

Because the performance analysis entails the behavior of objects used in our application code, following sections are devoted to a detailed description of these objects.

3.1.1 Segment Object

Given the current value of the LTS object, the segment object determines the appropriate video frames or parts of audio frames to be fetched and played in the current fetch cycle. It then schedules itself to be called back after a lapse of the *cycle time*. The application code also can set a parameter *send ahead*, that specifies the time that the server is supposed to be ahead of the client.

In the process of being recalled, based on the time it was actually supposed to be called and the time it was called, the video segment determines a number of frames to be dropped by using an *inverse binary order* [Smi]. This scheme attempts to minimize the consecutive dropping of frames, so that at the end, the stream would not appear *jittery*. In dropping the frames, the *mjpegSegment* also interpolates the display times of frames adjoining the dropped frame so that, although the frames were dropped, on the time scale no time gaps appear in the stream.

3.1.2 Play Object

In the application code we used, the play object checks and discards frames that are too late to be displayed, and then displays them in order. The video play object provides hooks to select the appropriate frame out of a collection that has been handed over to itself in a buffer.

3.1.3 Device Object

These objects are transparent to the programmer and contain low level calls specific to the platform and they submit data to be played, as has been called by the play object. The *SparcAuDevice* pre-empts *leftover* audio to *re-synchronize* at a regular frequency.

3.1.4 Network Twins

The network twins, *packet-Source (pktSrc)* and *packet-destination (pktDest)* uses a *cyclic UDP* protocol, where it is responsible for its own retransmission of the lost/delayed packets. This protocol has been explained in detail, with experimental verification of appropriateness for CM transport on best effort networks, in [Smi].

4 Experimental Evaluation of Synchronization Losses

As stated, the objective of our experiments is to measure the performance of CMT with respect to audio-video synchronization. In our performance evaluation, we do so with respect to metrics set-forth in Section 2, by computing the time drifts between corresponding video frames and audio samples.

4.1 Types of Experiments

We carried out two main experiments where, in the first one the segment object and the play object resided on the same machine, which we refer to as *local* objects. In the second experiment they were distributed across two machines. By the very design of CMT, this setup requires having *network twins*, the *packet source* and the *packet destination*. In the nomenclature of [MPR97], they are the *local playback application model* and the *simple remote playback application model*, of which the flow of streams are given in Fig. 2.

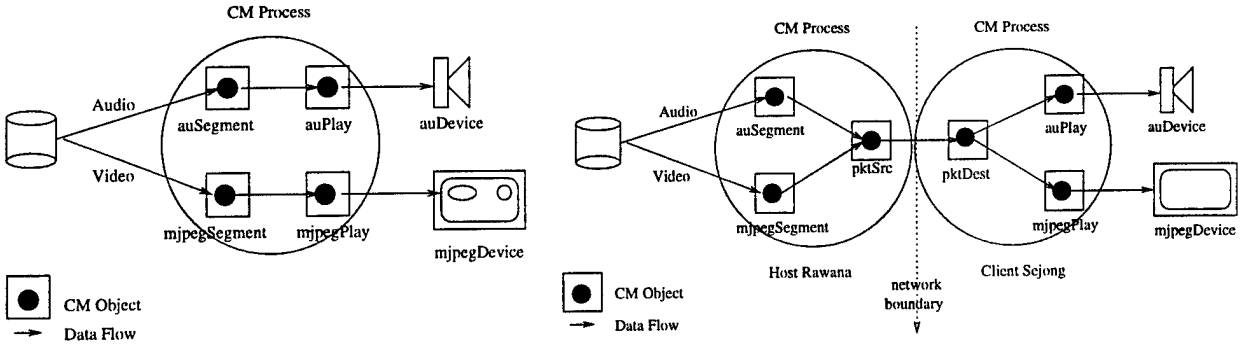

Figure 2: Local and Remote Playback Experiments in CMT

In our experiments we measured the performance of CMT with respect to selected metrics under varying processor loads and network loads. Accordingly, for local experiments we collected data under different processor loads and for remote experiments we collected data under different network and processor loads. In repeating experiments, the processor loads were measured by `top`, which gives the number of processes waiting for the processor, i.e. the length of the waiting queue. The experiments were carried out only when the number reported for the processor queue had stabilized for the immediately preceding 1, 5 and 15 minutes. The numbers reported are these

stable values. The network loads were measured by the use of a *network sniffer*, and the experiments were carried out under stable loads reported by the sniffer in terms of packets per second.

4.2 Experimental Parameters and Methodology

In all of our experiments, we displayed audio and video streams consisting of 1600 frames in each stream. We ran the local experiment with different processor loads of 0, 1 and 5, which we call *low*, *medium* and *high* loads. For network loads we ran the experiment with approximately 0, 200 and 2000 packets/second. For each load setting we repeated our experiment five times and collected average statistics over them.

Our experiments were carried out using two uni-processor Ultra Sparc 1's with 64 MB RAM each, connected by a 10 Mbps Ethernet. Our work stations were equipped with JPEG Parallax© cards, and we used the Motion-JPEG video format and the 8-bit SparcAu format for audio.

As stated in [MPR97], some care must be taken to ensure that the experimental methodology does not degrade the performance of the system, i.e. is as *non-intrusive* as possible. In this respect, we ensured that the time recorded by each module has been taken from already existing variables, and written into global arrays that were allocated before the system started to run in a stable state. Furthermore, these arrays were written into files when the objects were destroyed, thereby incurring minimal overhead in maintaining performance statistics. We also eliminated some initial readings to ensure that any start-up overhead does not affect final measurements. Finally, in order to keep track of frame numbers, we have introduced frame numbers into header portions of video and audio frames. Statistics maintained were the frame number and the time of arrival at each relevant CM object.

5 CMT's Performance on Steinmetz' Metrics

This section analyses our data with respect to Steinmetz' parameters. For each type of experiment (i.e. local with differing processor loads, remote with differing processor loads and remote with differing network loads) we show the progress of audio-video drift of a *typical* stream, and the average statistics over five runs. Finally, based on our data we present our conclusions.

5.1 The Local Experiment

As shown in Fig 2, segment, play and device objects were used. Their typical behavior with respect to Steinmetz' parameters and the average behaviors of all CM objects are given in Fig 3.

Outcome of the current experiment, as seen from Fig 3, indicates the following facts about CMT's performance:

1. In all CM objects (i.e. segment, play and device), imperceptible audio-video synchronization exists only for about 200 frames. Tolerable synchronization exists for about 400 frames.

2. Processor load increases audio-video timing drifts.

3. The *segment objects* have a considerable resilience in audio-video timing drift compared to *play* and *device* objects. We believe this behavior is due to the *inverse binary order* [Smi] based drops built into the video segment.

4. In order to provide audio-video synchronization, there needs to be a periodic re-synchronization mechanism in addition to LTS.

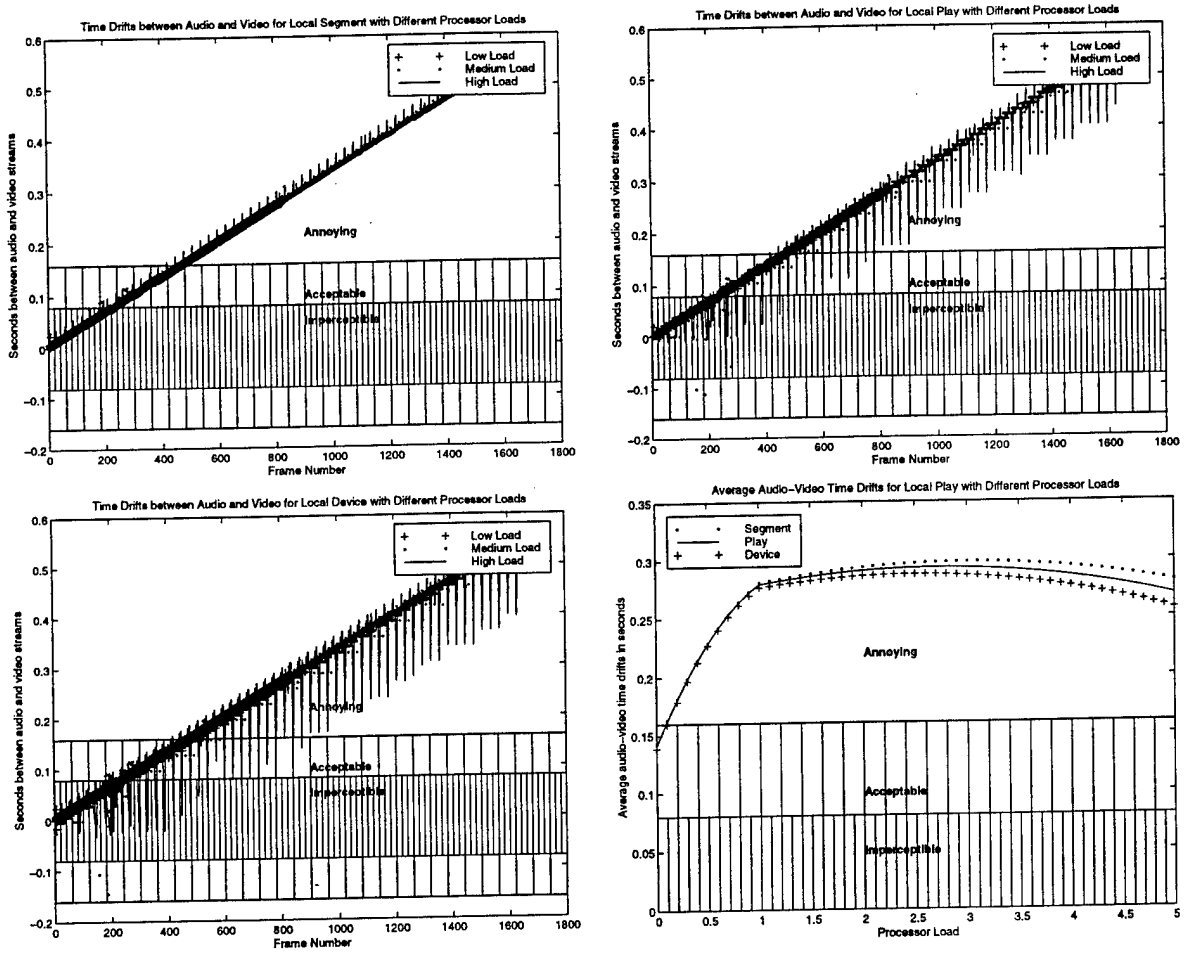

Figure 3: Synchronization Behavior of Local Playback

5. After an initial increase of audio-video synchronization drift, it appears to show some improvement. Although surprising at first, we soon noticed that under high load both streams drop a considerable number of frames, and thereby show some improvement in synchronization of the frame pairs that are present.

In order to verify our suspicion that the improvement in synchronization mentioned in conclusion 5, i.e. that the slight improvement in synchronization comes at the cost of stream continuity, we measured the frames drops in both streams. These are given in Fig 4.

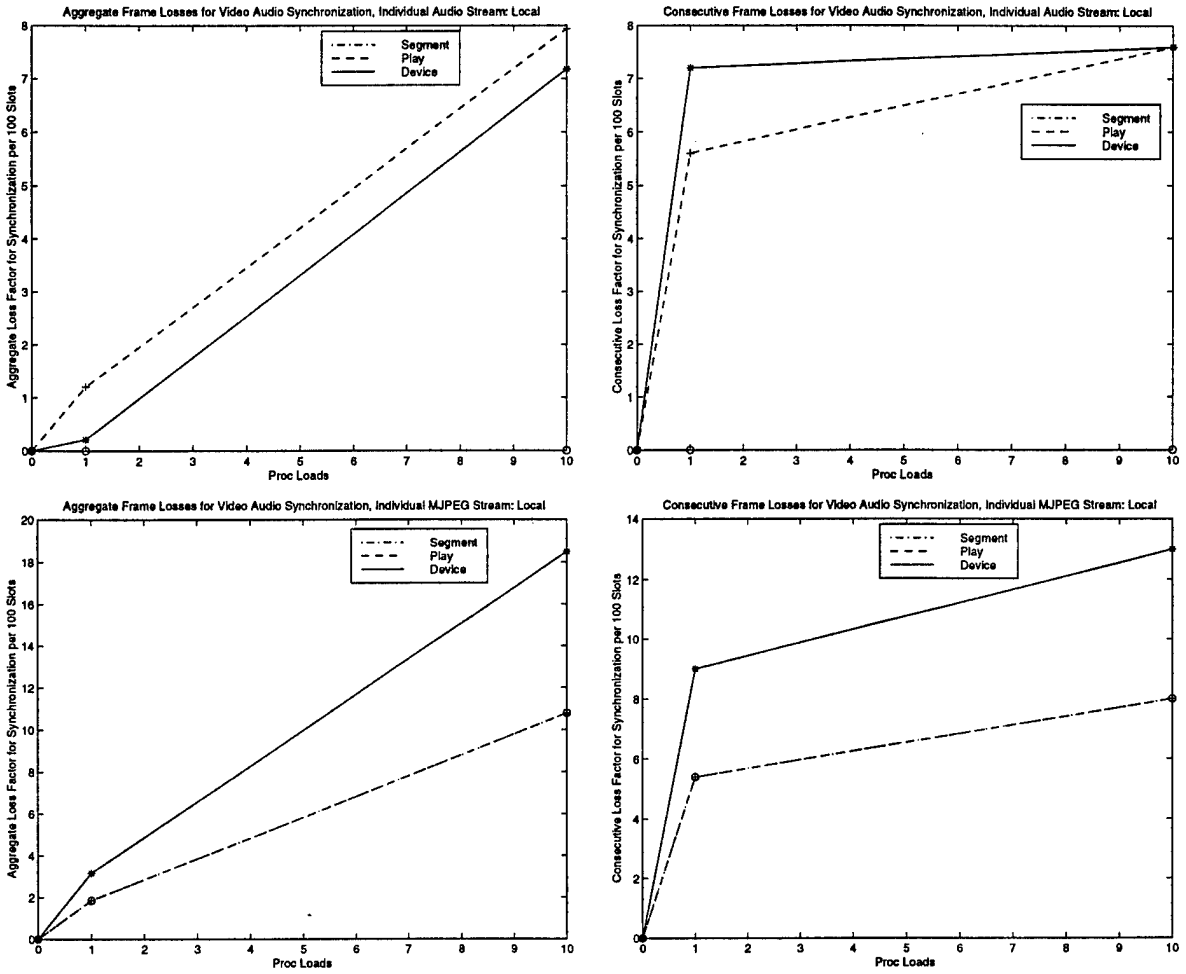

Figure 4: Frame Losses in Local Playback with Processor Loads

Based on the graphs for aggregate and consecutive losses in media streams, the following conclusions can be made:

1. The segment does not drop frames within the processor loads that we tested, although there is an in built mechanisms to do so, based on the delay in calling itself. Consequently, the processor loads do not cause such delays, and thus have no effect on the frame dropping mechanism built into the video segment.

2. Although the aggregate drops increase with processor loads, the increment in consecutive drops are minimal in both the play and device objects.

3. The play and device objects drop a significant number of frames, implying that they arrive *too late* to be played. The number of video frames dropped by the device is significantly higher than audio frames.

5.2 The Remote Experiment: Effect of Processor Loads

In this experiment, we subjected the simple remote playback to varying processor loads. These processor loads were generated by a seperate process on the same CPU, and applied to both the client and the server. As described in Section 5.1, the experiment involved segment, play, device, and the network twins packet source and packet destination. As in previous experiments, typical behavior of segment, play and device and their average behavior measured with respect to Steinmetz' metrics are given in Fig 5.

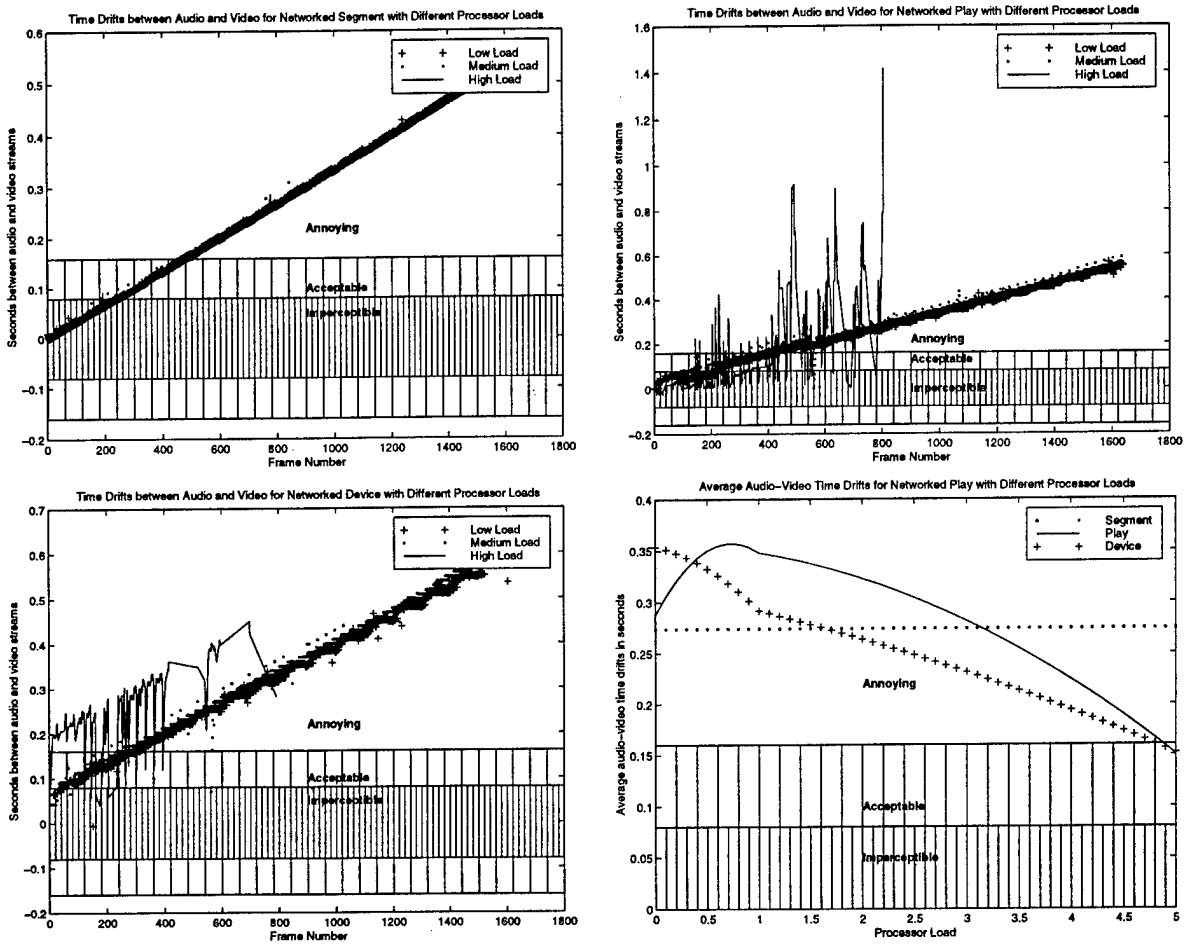

Figure 5: Synchronization Behavior of Remote Playback with Processor Loads

Outcome of the current experiment, as seen from Fig 5, indicates the following facts about CMT's performance over a local area network:

1. None of the CM objects have tolerable audio-video synchronization for any appreciable amount of time.

2. The segment object is not affected by the processor load as much as others.

3. There is vastly varying mis-synchronization between remote play and remote device objects during high processor loads.

4. The remote device and play objects appears to be gaining audio-video synchronization among the very few frames displayed on the average.

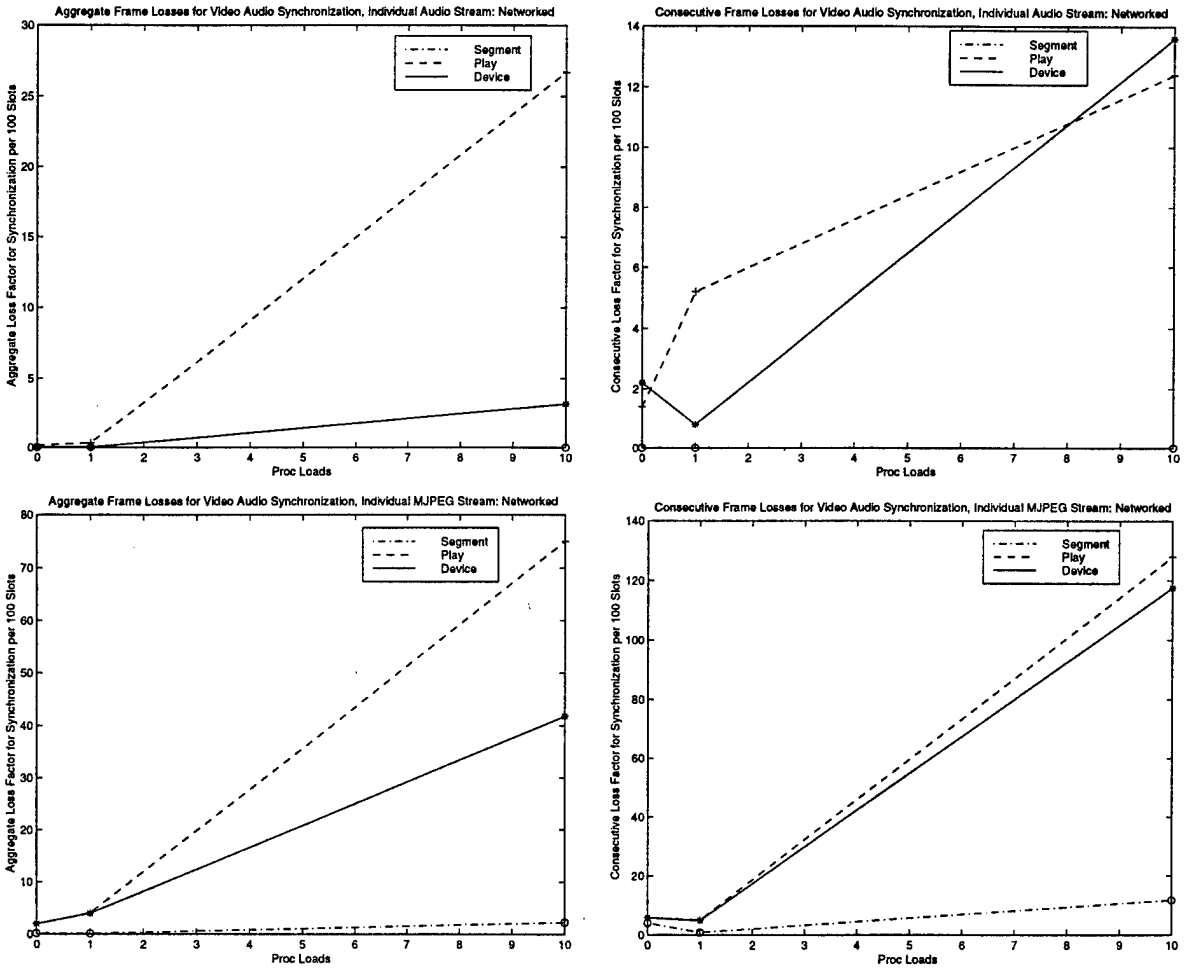

Figure 6: Frame Losses in Remote Playback with Network Loads

Based on the graphs for aggregate and consecutive losses in media streams, the following conclusions can be made:

1. The audio segment drops a constant amount of frames when there are processor loads both locally and remotely.

2. The frame drop procedure built into the video segment works remarkably well in the sense that the increase in video frame drops are kept to a minimum at the segment.

3. Due to the network, a significant number of frames, both audio and video are lost at both the device and play objects sitting at a remote client.

4. Consecutive loss in frame drops increases as the processor loads increase in remote play.

5.3 The Remote Experiment: Effect of Network Loads

In this experiment, we subjected the simple remote play to varying network loads. As described in Section 5.1, the experiment involved segment, play, device, and the network twins packet source and packet destination. As in previous experiments, typical behavior of segment, play and device and their average behavior measured with respect to Steinmetz' metric are given in Fig 7.

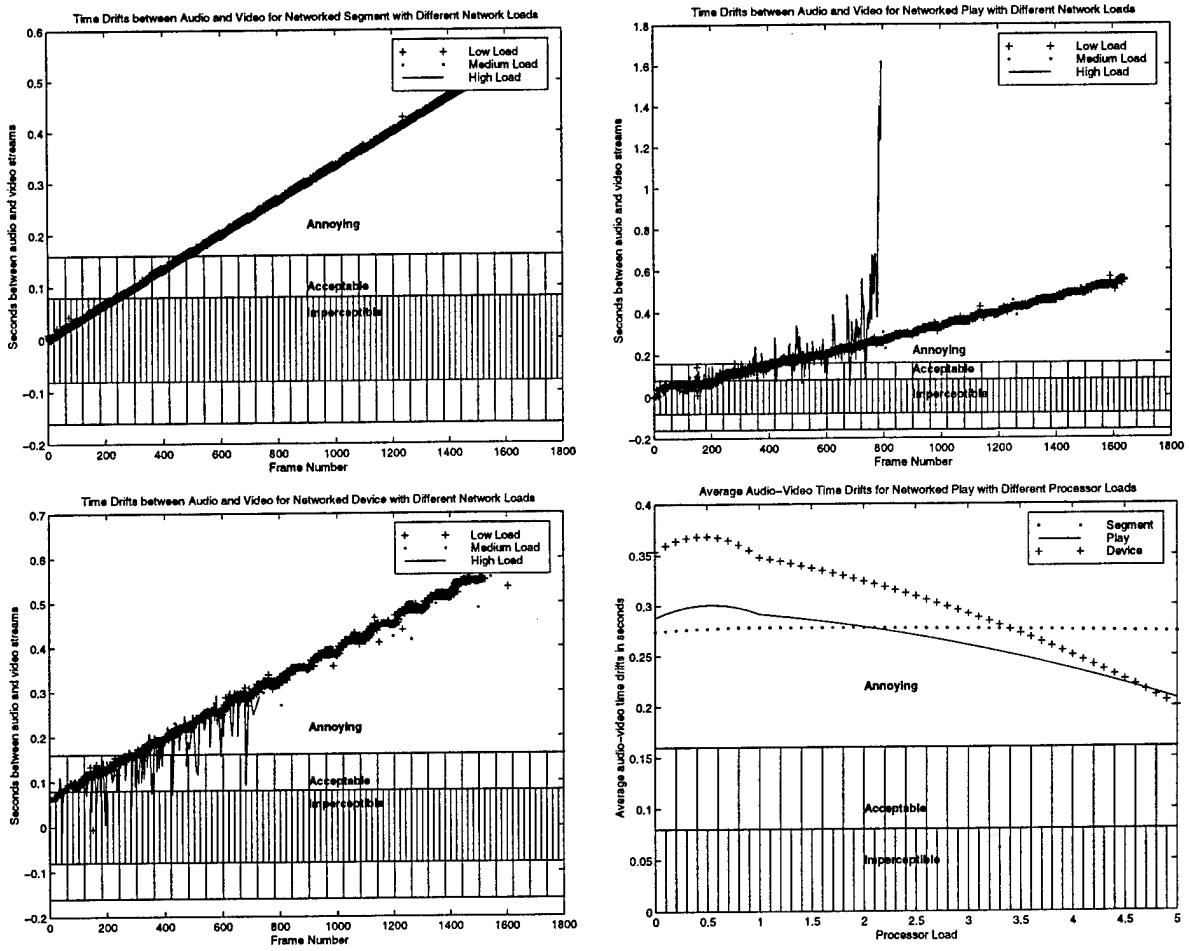

Figure 7: Synchronization Behavior of Remote Playback with Network Loads

Outcome of the current experiment, as seen from Fig 7, indicates the following facts about CMT's performance over a local area network:

1. Remote audio-video playback is not within humanly tolerable synchronization limits.
2. As the stream progresses, the audio-video mis synchronization increases without bound.
3. Segment objects are not affected by network loads. This is natural as they are at the server site, and do not depend on the network at all.
4. The variation of audio-video synchronization critically affects the play object, and consequently the device object. The degree of mis synchronization is much higher than the affect of high processor load. The actual number of frames dropped are given in Fig 8.

5. After some time, higher network loads result in loosing large amount of frames, so that the remaining frame pairs tend towards acceptable synchronization.

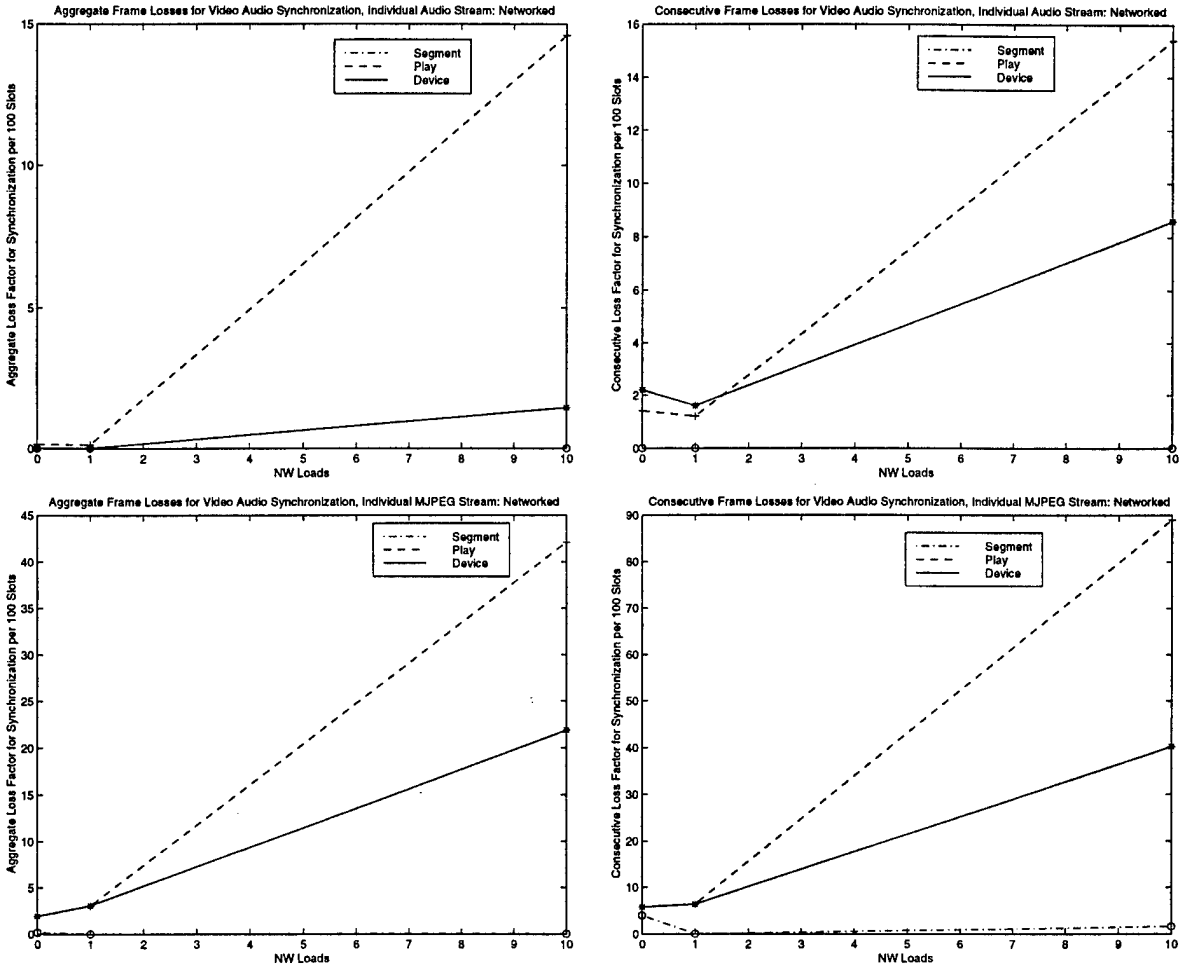

Figure 8: Frame Losses in Remote Playback with Network Loads

Based on frame drop graphs of Fig 8 and Fig 6, we notice that the effects of network load and processor load play a similar role in frame drops and maintaining synchronization.

5.4 Conclusions

Based on our experiments, when the client and the server objects are at the same site, audio and video are synchronized to be within tolerable limit in terms of Steinmetz' metrics only within the first 400 or so frames; i.e. first 13 seconds of playback. In the case of remote play, there is hardly any synchronization at all at the client site.

In order to maintain synchronization, for local clients and servers, some solution that incorporates periodic synchronization, such as [RR93, Sri92] may be required.

One of the major problems we encountered in applying Steinmetz' metrics to audio-video synchronization was the fact that the metrics were designed to be applied to synchronization between loss-less media streams, and those provided by CMT are lossy services. Consequently, we measured our traces with respect to lossy synchronization metrics described in Section 2.1.

6 CMT's Performance on Lossy CM Metrics

In this section, we analyze our data against synchronization metrics for lossy continuous media presented in Section 2.1.

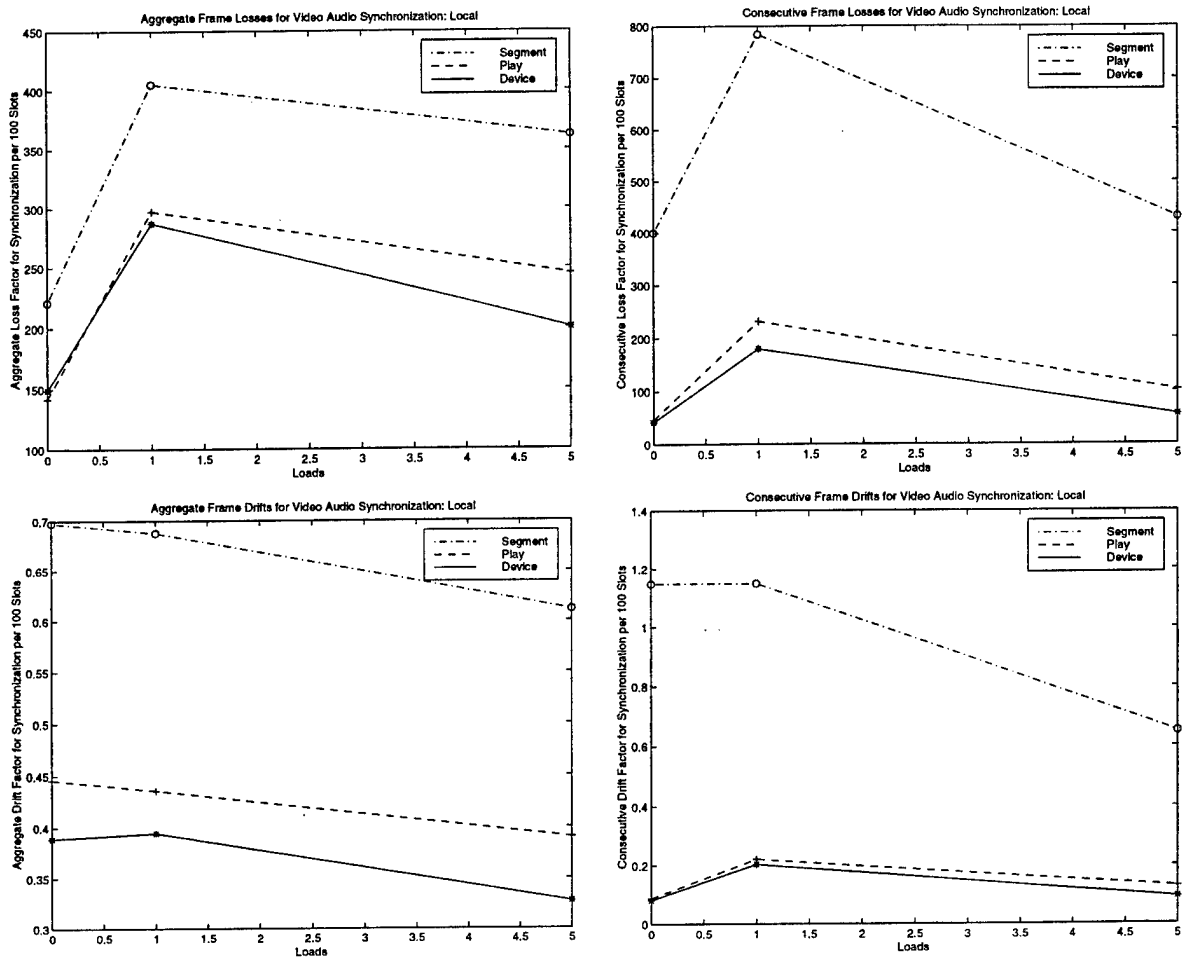

Figure 9: Synchronization Behavior of Local Playback

In these metrics, mis-synchronizations between audio and video streams were measured in terms of aggregate and consecutive losses of content, and aggregate and consecutive drifts. As described in Section 2.1, at the heart of these metrics lies the concept of *slots* to which media frames are suppose to belong. The difference in frame numbers belonging to a given slot measures a *unit content loss*, and aggregate and consecutive non zero such losses provide metrics of lossy synchronization. Time drifts are the aggregate and consecutive drifts within each such slot. Consequently, the so called *content losses* provide a coarse measure and drift parameters provide a finer measure. The values obtained for these metrics in our experiments are given in the following sections.

6.1 The Local Experiment: Lossy Metrics

The graphs in Fig. 9, viewed in light of the loss graphs of Fig. 4 and their comparison with those given for Steinmetz' metrics in Fig. 3, indicate the following:

1. From the very beginning there are aggregate synchronization losses between audio and video streams, although with processor loads they do increase. The difference here is more pronounced than in Steinmetz' metric. As the loads increase, the individual streams drop more frames, and consequently at the cost of loss of continuity, there is a marginal decrement in synchronization losses.

2. The device object always has a lower asynchrony, as it is passed only those frames that have arrived in time. This is evidenced in the higher aggregate losses for audio and video streams in Fig. 4, and in the aggregate synchronization graphs for devices of both streams given in Fig. 9.

6.2 The Remote Experiment: Effect of Processor Loads

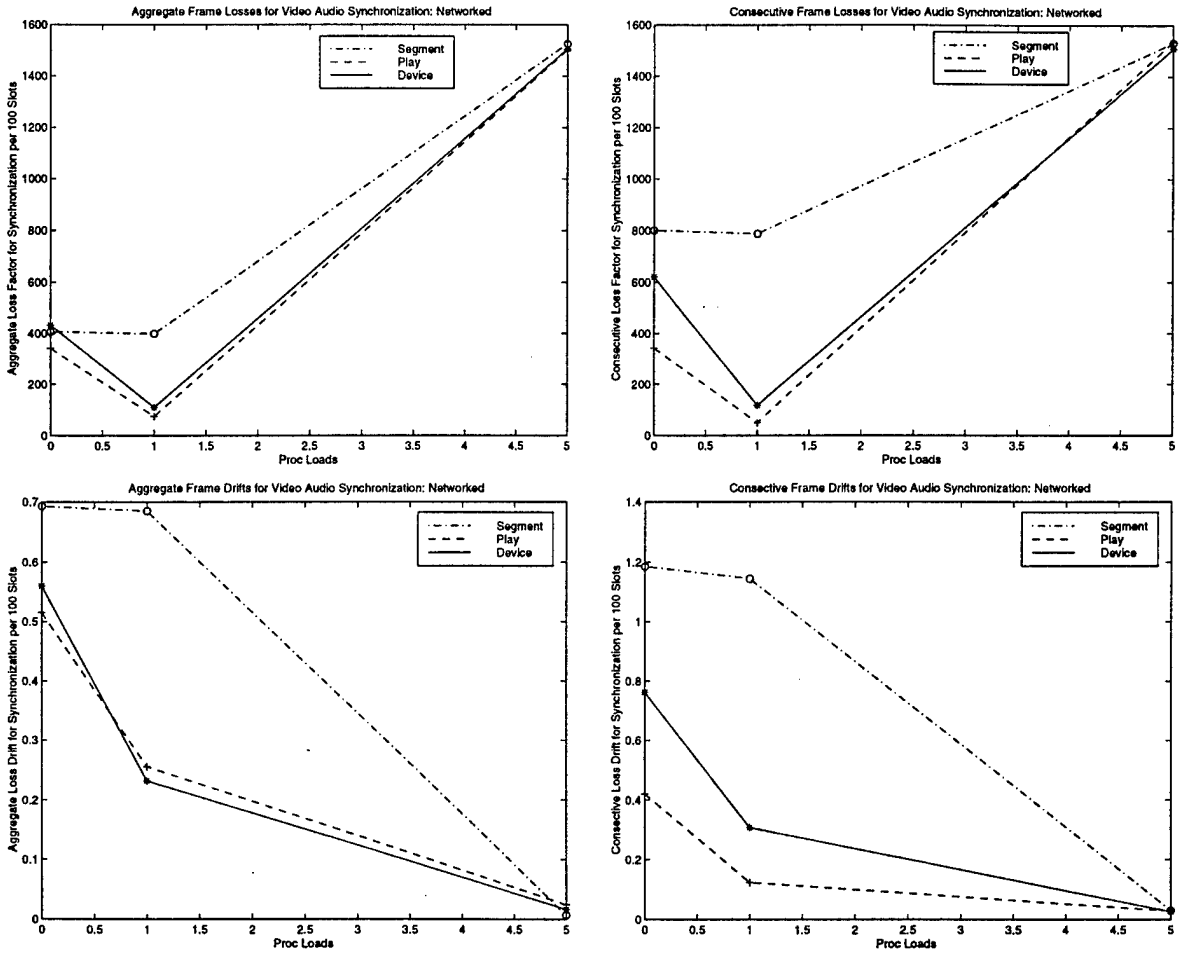

Figure 10: Synchronization Behavior of Remote Playback with Processor Loads

From the results of Fig. 10, we can draw the following conclusions.

1. As the processor loads increase, the loss of synchronization at segments, play objects and devices increase, and the increments are more than the corresponding increments in the local play.

2. From Fig. 6, we notice that aggregate frame losses remained constant over processor loads. Nevertheless, according to Fig 5, synchronization losses remained high, but constant. But according to Fig 10, the aggregate synchronization loss factor (ASLF) in Section 2.1 increases. The reason is that because the drops in both segments are un-coordinated, and only the pairs that are shown have approximately the same time gaps, Steinmetz' metric does show a constant value. But ALSF shows the loss of synchrony caused by un-coordinated drops between the two streams.

6.3 The Remote Experiment: Effect of Network Loads

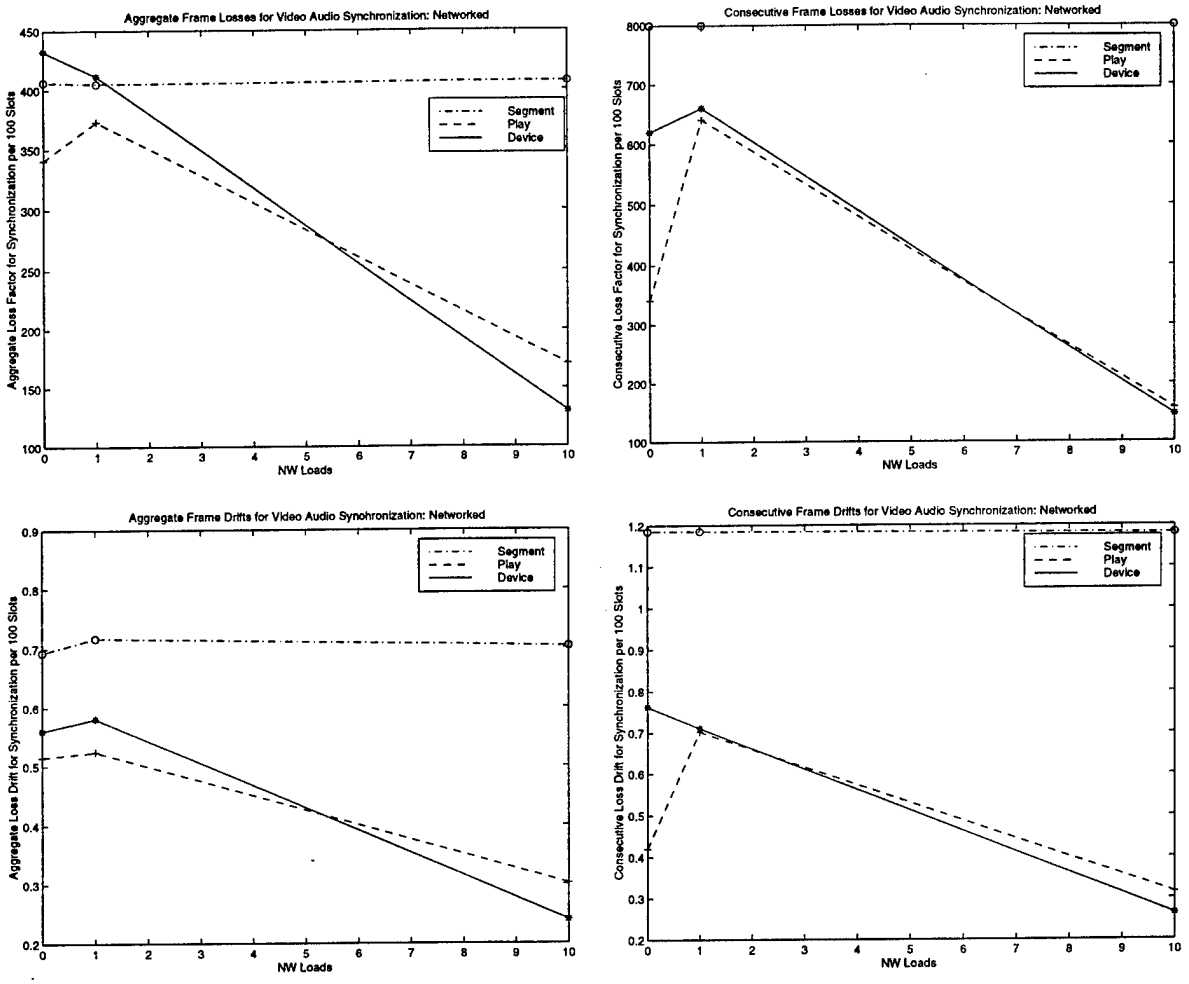

Figure 11: Synchronization Behavior of Remote Playback with Network Loads

Graphs of Fig. 11 indicate the following about the behavior of remote playback in the presence of network load.

1. The aggregate synchronization losses in the segment remains constant. Also, from Fig. 6 audio was not dropped, and video was dropped at a constant rate. Therefore, the loss between the two streams were *un-intentionally coordinated*, resulting in a constant ASLF over network loads. This is expected since the network should not affect the server, rather only the client (if at all). Rightfully, according to Fig. 7 Steinmetz' metric also indicates this trend.

2. As the graph indicates, the client side is badly affected by the network load, and both at the device and the play objects there is a remarkable drop in ASLF as the network load increases, which is somewhat surprising. Nevertheless, these values are far beyond the acceptable parameter ranges reported in the human perception study of [WSNF97].

6.4 Conclusions

Even by the lossy metrics of [WS96], as in the case of Steinmetz' metrics, synchronized play of audio-video streams in CMT quickly reaches outside the bounds of human tolerance. Consequently, as stated in Section 5.4, there is a need for synchronization services on top of CMT.

One of the problems we encountered in subjecting the data from our CMT experiments to the loss media metrics is the fact that when the *mjpegSegment* intentionally drops packets, it also interpolates the beginning and end times for video frames. This has two effects on our experiments. Firstly, it violates the uniform time length of slots as the metric assumes, thereby casting doubts of the applicability of these metrics to measure CMT's services. Conversely, we are unaware of any metric that can be applied to measure lossy synchronization where the slotting is allowed to be varied by the service provider at will. Similarly, we have mentioned the problems encountered in applying Steinmetz' metrics, as they are not designed to measure lossy synchronization.

Secondly, and more importantly, it destroys the ability to precisely synchronize downstream, as the new time stamps are not exactly what can be used to synchronize.

7 Concluding Remarks and Future Work

In a series of experiments, we have investigated the synchronization services provided by CMT. While it is the best continuous media programming testbed we are aware of, to provide synchronized audio-video services that are to be within humanly tolerable limits, there need to be additional mechanisms.

In our experiments we subjected CMT's audio-video services to metrics designed and tested for human tolerances of mis-synchronizations by Steinmetz'. We found that for a single site play there is imperceptible mis-synchronization only for about 10 seconds, and tolerable display only for about three seconds more. Furthermore, for a client and server connected by a relatively uncongested Ethernet, there seems to be no tolerable synchronization for any appreciable time limit at all. Because CMT drops media streams, and Steinmetz' metrics assumed that media streams were loss-less, we analyzed the same results with respect to another set of metrics and arrived at the same result. We have also encountered problems with applying these metrics to CMT data as the built-in drop policy for video alters the time stamps in video streams. In addition, this policy is likely to interfere with any time based re-synchronization that may be attempted by other objects downstream.

We believe that synchronization problems in CMT arise due to the lack of coordination between the two objects providing the same service for the two supposedly synchronized streams. At the implementation level they are woken up by two different events, and follow their own cycles for CM services. Also, in the network transmission there is no effort to coordinate the two streams together. We are currently working on providing a synchronization service for CMT as a partial solution to these problems [PVW+].

As stated, in applying Steinmetz' metrics to CMT data we encountered the problem of frame drops; and independently we have measured them under differing load conditions [VPW+97]. In future, we plan to make CMT drop frames based on user specified QoS thresholds.

References

[MPR97] Ketan Mayer-Patel and Lawrence A. Rowe. Design and Performance of the Berkeley Continuous Media Toolkit. In Martin Freeman, Paul Jardetzky, and Harrick Vin, editors, *Proceedings of Multimedia Computing and Networking*, pages 194–206, 1997.

[Ous94] John K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, Reading, Massachusetts, 1 edition, 1994.

[PVW+] Shwetal Parikh, Srivatsan Varadarajan, Duminda Wijesekera, Jaideep Srivastava, and Anil Nerode. Stream Groups: A QoS based Synchronization Support Service for the Continuous Media Toolkit. Working Paper at the University of Minnesota.

[RR93] S. Ramanathan and P. Venkat Rangan. Feedback Techniques for Intra-Media Continuity and Inter-Media Synchronization in Distributed Multimedia Systems. *The Computer Journal*, 36(1):19 – 33, 1993.

[SB96] Ralf Steinmetz and Gerold Blakowski. A media synchronization survey: Reference model, specification and case studies. *IEEE Journal on Selected Areas in Communication*, 14(1):5–35, 1996.

[SE93] R. Steinmetz and Clemens Engler. Human perception of media synchronization. Technical Report 43.9310, IBM European Networking Center, Heidelberg, 1993.

[Smi] Brian Smith. Cyclic-UDP: A Priority Driven Best-Effort Protocol. http://www.cs.cornell.edu/Info/Faculty/Brian_Smith.html/Publications.

[SNL95] Brian K. Schmidt, Duane Northcutt, and Monica Lam. A method and apparatus for measuring media synchronization. In T.D.C. Little, editor, *NOSDAV 95*, pages 190–201, September 1995.

[Sri92] Cormac John Srinan. *Synchronization Services for Digital Continuous Media.* PhD thesis, University of Cambridge, October 1992.

[SRS93] Brian Smith, Lawrence A. Rowe, and Yen S. Tcl distributed programming. In *Proceedings of the 1993 Tcl/Tk Workshop*, 1993.

[SRY93] Brian Smith, Larry Rowe, and S Yen. A Tcl/Tk Continuous Media Player. In *Proceedings Tcl 1993 Workshop*, June 1993.

[Ste96] Ralf Steinmetz. Human perception of jitter and media synchronization. *IEEE Journal on Selected Areas in Communication*, 14(1):61–72, January 1996.

[VPW+97] Srivatsan Varadarajan, Shwetal Parikh, Duminda Wijesekera, Jaideep Srivastava, Anil Nerode, and Mark Foresti. Experimental evaluation of media losses and time drifts in the continuous media toolkit. In *Eighth International Workshop on Research Issues in Data Engineering: Continuous-Media Databases and Applications - submitted*, 1997.

[Wel95] Brent B. Welch. *Practical Programming in Tcl and Tk.* Prentice Hall, New Jersey, 1 edition, 1995.

[WS96] Duminda Wijesekera and Jaideep Srivastava. Quality of Service (QoS) Metrics for Continuous Media. *Multimedia Tools and Applications*, 3(1):127–166, September 1996.

[WSNF97] Duminda Wijesekera, Jaideep Srivastava, Anil Nerode, and Mark Foresti. Experimental Evaluation of Loss Perception in Continuous Media. *submitted to ACM Multimedia*, July 1997.